Policy Framework for the Next Generation Platform as a Service

Kentis, Angelos Mimidis; Ollora Zaballa, Eder; Soler, José

Published in:
Proceedings of 27th European Conference on Networks and Communications

Link to article, DOI:
10.1109/EuCNC.2018.8443260

Publication date:
2018

Document Version
Peer reviewed version

Policy Framework for the Next Generation Platform as a Service

Angelos Mimidis, Eder Ollora, José Soler
DTU Fotonik
Technical University of Denmark
Kgs. Lyngby, Denmark
{agmimi, eoza, joss}@fotonik.dtu.dk

***Abstract*— The Platform-as-a-Service (PaaS) model, allows service providers to build and deploy their services following streamlined work-flows. However, platforms deployed through the PaaS model can be very diverse in terms of technologies and involved subsystems (e.g. infrastructure, orchestration). Thus, the means for deploying and managing a service can significantly vary depending on the deployed platform. To address this issue, this paper proposes a policy-based framework designed for the Next Generation Platform-as-a-Service (NGPaaS). This framework allows service providers to define platform-wide and technology-agnostic policies to NGPaaS, by means of abstraction of the underlying platforms and the use of generic interfaces. The paper also presents a specific use case for the proposed framework, which targets network-oriented policies.**

Keywords— SDN, NFV, Policies, Policy Enforcement

I. INTRODUCTION

Recent paradigms like Software Defined Networking (SDN) and Network Function Virtualization (NFV), have transformed the networking industry. These technologies allow for better automation in resource allocation and management, faster service provisioning and the like. This has resulted in new and innovative network services (e.g. mobile edge computing virtual reality), that can profit from these advantages. However, SDN and NFV operate in different subsystems (networking for SDN and compute for NFV), thus the synergy between them is limited. To address this issue a number of orchestration platforms have been developed, mainly based on the ETSI-Management and Orchestration (MANO) architecture [1]. These platforms allow SDN and NFV to complement each other. However, the architecture of these platforms is complex; and developing custom solutions to fit specific use cases (e.g. an Internet of Things (IoT) centric platform) is a very difficult task. Recently, there has been interest in designing and developing frameworks that can automate the platform provisioning process by translating business-cases to specific platform configurations. This way, provisioning build-to-order platforms is faster and more efficient. One such framework is the Next Generation Platform as a Service (NGPaaS) [2].

Given the nature of NGPaaS and depending on the selected business-case, a service can be deployed in a very technology-diverse set of platforms. This implies that the means, by which the service is managed, will also vary. In addition, a service deployed through NGPaaS can span across multiple subsystems (compute, network, orchestration etc), which greatly increases management complexity.

A verified approach for service management in platforms like NGPaaS is through policy enforcement. To that end, this paper proposes a mechanism which facilitates the definition, evaluation and enforcement of policies in every subsystem of NGPaaS. This mechanism will allow service providers which are tenants of NGPaaS, to define their requirements, to the different platform/service deployments in the form of policies. These policies will be generic in nature, thus abstracting the underlying technologies and subsystems. The policy mechanism will then push these generic policies into NGPaaS. As policies proliferate through NGPaaS they will be translated to subsystem and technology specific policies. For the remaining of the document this policy related mechanism will be termed as a policy framework, since it functions as a support system to NGPaaS. The paper is structured as follows; Section II outlines the necessary background and related scientific literature. Section III describes the architecture of the proposed policy framework; by first listing the requirements it needs to meet and then by mapping these requirements into specific characteristics. Section IV presents a specific use-case with focus on network policies, which aims to highlight the expected behavior of the policy framework. Finally, Section V provides a conclusion.

II. BACKGROUND AND RELATED WORK

A. Multi and single subsystem policy frameworks

As described in Section I, most modern service platforms [3] [4] span across multiple subsystems. The *Infrastructure Subsystem* includes the (physical or virtual) computing, storage and networking resources. The *Infrastructure Control Subsystem* includes the technologies that offer control over the *Infrastructure Subsystem* (e.g. an SDN Controller). The *Orchestrator Subsystem* coordinates the compute, storage and networking subsystems, allowing for joint control of the available infrastructure. Finally, the *Business Subsystem* encompasses the Operations Support System (OSS) and Business Support System (BSS). These subsystems are illustrated in Fig. 1.

978-1-5386-3873-6/17/$31.00 ©2017 IEEE

Due to this multitude of subsystems in which policies can be applied to, a policy framework can be implemented in one of two ways. The first way is to implement isolated policy frameworks for each of the subsystems; the second is to implement a single framework that integrates all the individual subsystems. The latter offers a greater degree of control over the infrastructure, as it facilitates the definition and management of multi-subsystem policies. Additionally, it provides easier administration, since there is a single point of reference for the policy management of the whole platform. However, this introduces complexity in the framework's design and an overhead in inter-subsystem communication.

B. The Next Generation Platform as a Service

The scope of NGPaaS [2] is to provide the means of translating business cases into specific platform deployments. As a result, service providers can focus on the services themselves and not in the means through which the services and platforms are deployed and managed. NGPaaS comprises a multi-layered architecture, with each layer including one or more subsystems. The *Business as a Service (BaaS)* layer comprises the business subsystem of NGPaaS and supports multi-PaaS operation and control functions. The *Platform as a Service (PaaS)* layer comprises the orchestrator and infrastructure management subsystems. The *Infrastructure as a Service (IaaS)* layer comprises the Infrastructure subsystem. Finally, the *Dev-for-Operations* layer, which is an evolution of the traditional DevOps model. However, for the scope of this work, it is not strictly relevant and it is therefore excluded. These layers are illustrated in Fig. 1, in parallel with the different subsystems. Thanks to this modular architecture, NGPaaS can facilitate the overall platform/service deployment and management and easily integrate new services and platforms allowing for quick time-to-market, when compared to the traditional service/platform deployment paradigm.

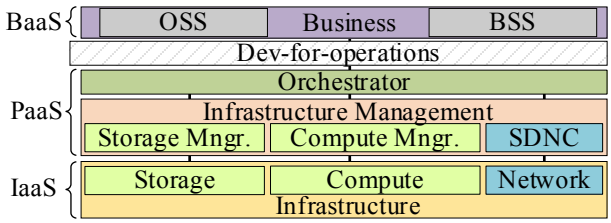

Fig. 1. The different platform subsystems of a cloud platform (right) and how they associate with the different layers of NGPaaS (left)

C. Policy Core Information Model (PCIM)

To address the challenges related to managing platforms like NGPaaS, a new policy model is needed. The authors of [5] present an object-oriented information model in order to represent policy-related information in a low-level, vendor- and device-independent manner. The information model is divided into two hierarchies of object classes. (1) The *structural classes*, which represent policy information and control of policies. And (2), the *association classes* which indicate the relation between instances of structural classes. Although the initial application of the information model was expected to be for Quality of Service (QoS) related policies, the policy classes and associations are generic enough to accommodate policies related to a variety of topics (e.g. security, energy efficiency, etc). Therefore, network administrators, software developers or policy administrators could represent a big variety of policies using the same base design of policy objects.

The PCIM was later extended in [6], which included new elements like the extensions for header filtering. Although there are several changes and extensions to the previous specification, the document provides interoperability with the original PCIM model implementation.

D. Policy-based resource management for NFV

The work in [7] and later in [8] addresses the procedures for resource management and workload distribution in NFV infrastructures. It tackles the constraints of compute and network capacity of NFV Points-of-Presence (PoP). That document also presents the conceptual difference between global and local policies. A global policy engine may enforce global policies while subsystem-specific policy engines may only enforce local policies. The authors also point out that the policy framework is hierarchical in nature, since the policy engine that belongs to a certain subsystem may be part of a broader subsystem. Another important concept addressed by that work is the need of a Publication/Subscription (PUB/SUB) bus so that the different entities in the MANO architecture can subscribe for policy updates.

Moreover, the ongoing work in [9] defines an architectural framework for policy-based resource management (in terms of placement and scheduling). The framework is based on the architecture defined in the documents previously analyzed in the section. The policy engine would interface with the measurement entities so that periodical resource utilization statistics are retrieved and a better resource placement and scheduling are achieved.

E. Policy frameworks for network resource management

The authors in [10] propose a generic datacenter management framework that encompasses the policy-based management of the data, control and orchestration of functional entities. The authors state that the proposed framework focuses on two mains aspects: (1) a unified policy definition and enforcement methodology and (2) an increased extensibility using a loosely coupled model-driven architecture. The policy framework proposed is implemented following the model proposed in [5], [6]. Besides, a policy state machine is also described. The state machine supports the different states in which the policies could be, depending on the validation stage (conflict validation, variable resource allocation…). The policy framework architecture presented in this research paper depicts the inclusion of the OpenDaylight (ODL) SDN controller [11] to install network policies. The validation of the policy framework implementation shows that there is no considerable time difference between policy installations. The number of installable policies did not have a great impact on installation time. However, it is noted that

more complex policies are expected to take more time to install due to increased complexity in the validation steps.

In [12] the researchers propose another policy management framework that addresses several challenges arisen from a highly decentralized and miscellaneous environment. This framework highlights several of the challenges of policy-based management in the context of future networks. One of the outlined challenges is the need for a transformation process of the operator's high-level objectives to network-understandable policies. This is achieved by a Human-To-Network (H2N) interface. The work also addresses the need for concepts like Policy Continuum and Policy Translation. Policy Continuum ensures that policies written in relative and high-level terminology can be transformed into lower level policies with different corresponding features. Policy Translation lies in the detection and specification of high-level information into the targeted level information. Furthermore, a common information model would help on the comprehension of the transmitted policies. As the main outcome, the paper presents a policy framework to enable autonomic service and network management and address relevant policy-related challenges.

Compared to the related work presented in this section, the novelty of this paper is two-fold. (1) It is designed with the NGPaaS in mind, making it timelier and future-proof proposal and (2) It encompasses the whole PaaS and is not limited to a specific subsystem.

III. ARCHITECTURE OF THE FRAMEWORK

This section of the paper presents the architecture of the proposed policy framework. As a first step, some requirements will be highlighted to better communicate the objectives that the framework attempts to fulfill.

A. Requirements

The first requirement (**R1**) is that the policy framework should operate across all the subsystems of the platform. Despite the associated implementation complexity, an architecture based on this premise will allow better overall control of the available infrastructure. Second (**R2**), the framework should allow policies to be defined in generic terms, not associated with specific technologies. Third (**R3**), the framework should be easily extensible with new subsystems and policies. Fourth (**R4**) policies should be defined in a human-friendly format. Finally (**R5**), all subsystems should use a common model to express the policies and their lifecycle. Doing so will facilitate the definition of multi-subsystem policies.

B. How policies are modeled

Within the proposed policy framework, policies are loosely modeled based on the PCIM [5][6]. Each policy is represented by a *Policy Rule,* which comprises a set of *Policy Conditions* and *Policy Actions.* A set of *Policy Conditions* can be defined in either a Conjunctive Normal Form (CNF) or a Disjunctive Normal Form (DNF). CNF-based *Policy Conditions* are expressed as an AND set of ORs (e.g. [A OR B] AND [C OR D]), while DNF-based *Policy Conditions* are expressed as an OR set of ANDs (e.g. [A AND B] OR [C AND D]). Each singleton *Policy Condition* comprises a pair of a *Policy Variable* and a *Policy Value.* If the *Policy Variable* matches the *Policy Value,* the *Policy Condition* evaluates to true, else to false. Following the same premise if the CNF or DNF-based *Policy Conditions* evaluate to true, then the associated *Policy Rule* is considered enforceable. In order for a *Policy Rule* to be enforced, the *Policy Actions* associated with it must be applied. *Policy Actions* also comprise a pair of a *Policy Variable* and a *Policy Value. Policy actions* can be declared either as ordered or not. Ordered *Policy Actions* MUST be applied in a specific order, whilst non-ordered *Policy Actions* can be applied without specific ordering. An action is considered applied when the *Policy Variable* meets the associated *Policy Value.* A *Policy Rule* is considered enforced when all the associated *Policy Actions* have been applied. To facilitate conflict resolution, *Policy Rules* can also be assigned priorities, so in case of two (or more) conflicting *Policy Rules* only the one with the highest priority will be applied. Finally, to allow for better management of the infrastructure, multiple *Policy Rules* can be aggregated into *Policy Groups.* These groups can then be managed as a single entity by the policy framework. Fig 2 illustrates the policy model.

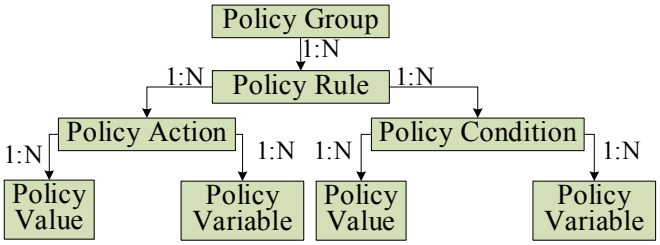

Fig. 2. The policy model utilzied by the policy framework

C. Lifecycle of policies

In order to be properly managed by the policy framework, policies exist within a specific lifecycle illustrated in Fig. 3. Upon arrival to the policy framework (1), all policies start in the *New Policy* state, meaning that the framework has not yet evaluated if their definitions match any of the policy definitions known by the policy framework. If they fail this formal validation (2), policies move to the *Removed* state and are purged from the policy framework. If they pass formal validation (3), then they move to the *Formally Validated* state. This state implies that the framework has a notion of how to enforce the policy, but has not yet evaluated if the policy conflicts with other existing policies. If the policy fails this conflict resolution (4), it is moved to the *Pending* state. This state is for policies that the administrator has expressed intent to enforce, but for one reason or another they cannot yet be enforced in the infrastructure. If the policy passes conflict validation (5), it is then moved to the *Conflict Validated* state. In this state, the framework has not evaluated if the infrastructure can accommodate the policy, in terms of available resources. If the policy fails this context validation

(6), it moves to the *Pending* state, waiting for the proper circumstances to arise (7). If the policy passes context validation (8), it is moved to the *Enforced* state. At any point in time, the administrator can request a policy to be removed (9), which means that the policy will move to the *Removed* state and will be purged from the framework. Finally, the administration might request an enforced policy to be deactivated (10), at which point the policy will transition to the *Pending* state waiting to be activated again (11).

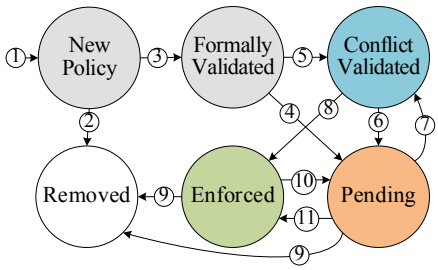

Fig. 3. The lifecycle of policies within the proposed framework

D. Architecture of the policy framework

Building upon the multi-subsystem platform of NGPaaS and the aforementioned requirements from section III.A, this paper proposes the policy framework architecture of Fig. 4.

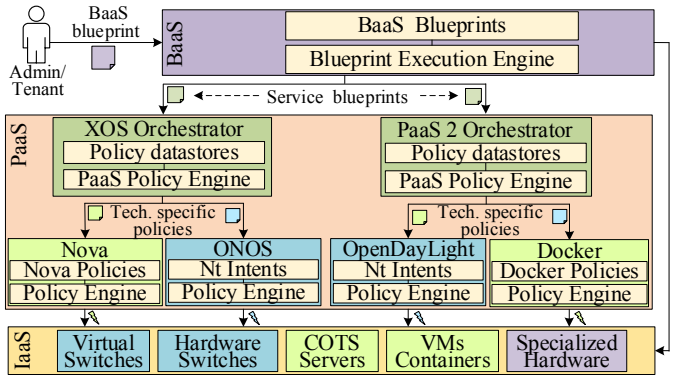

Fig. 4. Architecture of the policy framework

There, a centralized BaaS layer is responsible for the management and orchestration of the different PaaSes under its control. These PaaSes can belong to the same or different tenants of NGPaaS. The means through which the different PaaSes are instantiated is out of the scope of this paper, but is described in detail in [2]. The policy framework allows generically defined policies (i.e. not bound to specific subsystems or technologies) to proliferate from the BaaS layer, through the different PaaSes and to the underlying infrastructure. Each entity (BaaS, orchestrator, SDN Controller (SDNC), etc), has its own implementation of repositories in which to hold policies and policy definitions. In addition, each entity also holds a policy engine that is responsible for managing the lifecycle of its associated policies and for decomposing and propagating policies to lower subsystems (e.g. from the orchestrator to the SDNC).

All policy requests must be first addressed to the BaaS layer, as this allows for a single point of management for NGPaaS and for policies that can span across multiple PaaSes. With regards to the BaaS layer, policies must be defined in the form of *BaaS blueprints* which must conform to one of the available *BaaS blueprint* templates provided by the BaaS layer. Upon receiving a *BaaS blueprint* the BaaS layer will decompose it (using the *Blueprint execution engine*) into one or more *Service blueprints*. Each *Service blueprint* consists of policies that are specific to a single service and a single PaaS. Since the BaaS layer has a global view of the deployed PaaSes and their technologies, it will be able to translate the generic policies defined in the *BaaS blueprint* into PaaS specific policies (using the *Blueprint execution engine*). Before decomposing the BaaS policy into PaaS policies, the BaaS layer will also perform the high-level formal, conflict and context validation of the generic policy defined in the BaaS blueprint. Doing so will stop the proliferation of the policy to lower subsystems in case of a conflict or misconfiguration, removing unnecessary processing in the lower subsystems.

The individual *Service blueprints* will then be sent to the corresponding PaaS orchestrators. In order to facilitate a universal messaging platform between the BaaS and the orchestrators, each orchestrator will need to be extended with an Application Programming Interface (API) translation layer. This layer will translate NGPaaS specific to orchestrator specific API calls. As with the BaaS layer, each orchestrator will also require the mechanisms to (1) store policies and policy definitions and (2) manage the lifecycle of its policies. The first mechanism can be implemented in the form of *policy datastores*. The specific technology with which the *policy datastores* are implemented will depend on the technologies offered by the corresponding orchestrator. The second mechanism can be implemented in the form of an orchestrator specific *policy engine*. The role of the *policy engine* will be twofold; (1) Decompose the *Service blueprints* into policies for the orchestrator and policies for the infrastructure management subsystem. (2) Manage the lifecycle of the orchestrator related policies. As with the BaaS layer, the orchestrator can also stop the proliferation of policies to the underlying subsystems, if it assesses that a specific policy is unenforceable (e.g. due to conflicts with other policies). In any other case, the orchestrator will send the decomposed policies to their corresponding infrastructure management entities (e.g. the SDNC). Since the orchestrator has a global view of its platform, it is aware of the associated technologies that comprise the infrastructure management subsystem. This allows the orchestrator to translate policies for the infrastructure management subsystem from generic to technology-specific (through the *PaaS policy engine*).

Upon receiving a policy from the orchestrator and verifying that it can be enforced (valid policy, no conflicts and sufficient resources), the infrastructure management entities (e.g. the compute manager) will translate the policy into specific actions and enforce them in the infrastructure.

A policy, as defined by an NGPaaS tenant or administrator, can end up decomposed into multiple policies

(e.g. for different platforms and/or subsystems). As a result, its lifecycle cannot be managed by a single state machine, but instead, it will be managed by a set of state machines spread across NGPaaS. If all state machines reach the enforced state, then the policy can be considered as enforced in the infrastructure. For example, if the orchestrator of a PaaS assesses that a policy does not conflict with any other policies under its control, it will pass the policy to the associated network and compute managers. However, until it receives an acknowledgment from both that the policy has been enforced, it should keep the policy state as pending.

IV. Use Case for Network Policies

To facilitate the understanding of the proposed policy framework, this section will provide a specific use case with a focus on network policy enforcement.

An NGPaaS tenant, which is the owner of an already deployed PaaS (PaaS 1), wants to provide connectivity between two hosts (Host1 and Host2). PaaS1 comprises XOS [13] as the orchestrator and of ONOS [14] as the SDNC. As a starting point, the NGPaaS tenant will configure and send a BaaS blueprint with the desired configuration parameters to the BaaS layer. Since the NGPaaS tenant does not have knowledge of the underlying technologies that compose its platform, it will define the policy using generic terms. For example, the policy could be defined in CNF as: [(Host1 eq. Online) AND (Host2 eq. Online)] THEN [Connect(Host1,Host2)]. Once the BaaS blueprint is received at the Baas layer, the Blueprint Execution Pipeline will examine the blueprint and decompose it into a Service blueprint for PaaS 1. Since the orchestrator of PaaS 1 is XOS, the BaaS layer will compose the service blueprint using TOSCA [15] and pass it over the associated REST API of XOS. Before sending the service blueprint to XOS, the BaaS layer will perform its own high-level formal, conflict and context validation of the policy. If the policy passes all checks it will move to the *Pending* state, waiting for the outcome of the XOS related policy. At this stage, XOS makes sure that the Service blueprint is also formal, conflict and context validated. Before the orchestrator policy is flagged as *Enforced*, XOS will decompose it into policies specific to the individual infrastructure management subsystems. Since the policy is only network related, it will be decomposed into a single network policy for the ONOS SDNC. This policy will then be sent to ONOS, so that it can be further validated and enforced into the infrastructure. Upon receiving the policy, and given that it passes the formal, and conflict validation, ONOS will decompose the network policy into *network intents* [16]. This will trigger a comparison (divergence analysis) between the actual and the intended network state (is there a path between Host1 and Host2?). If the intended network state can be served, ONOS will enforce the network intents, hence providing connectivity between Host1 and Host2. At this point, the policy state in ONOS will move to the *Enforced* state. Then, ONOS will inform XOS about this outcome, which will lead the policy in XOS to move from the *Pending* to the *Enforced* state. This process will also be performed between XOS and the BaaS layer. When the policies in all subsystems reach the enforced state, the BaaS layer will inform the tenant that its request has been fulfilled. In any other case, it will return an error message.

V. Conclusion

This paper introduced the concept of a policy framework for NGPaaS. The proposed framework can simplify the management of NGPaaS, by allowing the different actors (administrator, tenants) to express their requirements from their platform deployments using well-defined, platform-wide and technology-agnostic policies. It is the role of the policy framework to then decompose the generic policies into technology specific and subsystem specific policies and propagate them through the deployed platforms. In addition, the policy framework is also responsible for the lifecycle management of these policies. To provide a comprehensive example of the internals of the proposed policy framework, this paper also presented a use case focused on network policies in Section IV.

Acknowledgment

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programmes, under Grant Agreement No. 761 557 (http://ngpaas.eu).

References

[1] J. Quittek, P. Bauskar, T. BenMeriem, A. Bennett, M. Besson, and A. Et, "Network Functions Virtualisation (NFV); Management and Orchestration," *Gs Nfv-Man 001 V1.1.1*, vol. 1, pp. 1–184, 2014.

[2] A. Mimidis *et al.*, "The Next Generation Platform as a Service", 2018 25th International Conference on Telecommunications (ICT), Saint-Malo. Accepted, to be published.

[3] "CORD." [Online]. Available: http://opencord.org/.

[4] "Open Source MANO." [Online]. Available: https://osm.etsi.org/.

[5] B. Moore, E. Ellesson, and A. Westerinen, "Policy Core Information Model -- Version 1 Specification,", RFC 3060, 2001.

[6] B. Moore, "Policy Core Information Model (PCIM) Extensions", RFC 3460, 2003.

[7] W. Steven, N. Figueira, R. Krishnan, and D. Lopez, "Policy Architecture and Framework for NFV Infrastructures," Internet-Draft, 2015.

[8] R. Szabo, S. Lee, and N. Figueira, *Policy-Based Resource Management*. Internet-Draft, 2016

[9] N. Figueira, "NFVIaaS Architectural Framework for Policy Based Resource Placement and scheduling,", Internet Draft, 2016.

[10] C. Caba, A. Mimidis, and J. Soler, "Model-Driven Policy Framework for Data Centers (Short Paper)," *2016 5th IEEE Int. Conf. Cloud Netw.*, pp. 126–129, 2016.

[11] "OpenDaylight." [Online]. Available: https://www.opendaylight.org/.

[12] A. Galani *et al.*, "A policy based framework for governing Future networks," *2012 IEEE Globecom Work. GC Wkshps 2012*, pp. 802–806, 2012.

[13] "XOS." [Online]. Available: https://wiki.opencord.org/display/CORD/XOS%3A+The+CORD+Controller.

[14] "ONOS." [Online]. Available: http://onosproject.org/.

[15] D. Palma, T. Spatzier "Topology and Orchestration Specification for Cloud Applications - PRIMER,", 2013.

[16] "ONOS intent framework." [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework.